REMARKS

Applicant submits herewith an RCE to enter the above claim amendments, and respectfully requests reconsideration of the instant application in view of the following arguments.

The Final Office Action dated August 5, 2008 ("Office Action") rejected claims 1, 5-7, 13-17, and 19-22 under 35 U.S.C. § 103 as being unpatentable over Bates *et al.* (U.S. Patent Application Publication 2003/0221185 A1 ("Bates")) in view of Bates *et al.* (U.S. Patent No. 7,251,808 B2 ("*Bates_2*")), and rejected claims 2-4 and 8-12 under 35 U.S.C. § 103 as being unpatentable over *Bates* in view of *Bates_2* in further view of Dandoy (U.S. Patent Application Publication 2004/0230954 ("*Dandoy*")).

A. Claim 18 Should Be Allowed Because It Was Not Rejected in the Office Action

As a preliminary matter, the Office Action does not specifically reject claim 18. Because claim 18 was not specifically rejected, Applicant respectfully submits that the limitations of claim 18 distinguish this claim over the cited references, and, as such, this claim should be allowed.

B. The § 103 Rejections Should Be Removed Because the Cited References Do Not Teach or Suggest Each Element of the Claims

The Examiner has rejected claims 1-17 and 19-22 under 35 U.S.C. § 103(a) as explained above. Applicant respectfully submits that the references of record do not present a *prima facie* case of obviousness because these references, considered alone and in combination, do not teach or suggest each limitation of the claims, as explained in detail below.

C. *Bates* in view of *Bates_2* Does Not Established a *Prima Facie* Case of Obviousness for Independent Claims 1, 17 and 20-22

1. *Bates* and *Bates_2*, Considered Alone and In Combination, Do Not Teach An Expression Evaluator that Evaluates Attributes wherein the Evaluation Determines that a Value to Be Shown for a Filed or Property Is the Result of an Evaluation of an Expression and Is Different from the Value that Is Shown Without the Attribute as Claimed in Claims 1-22

Each of the independent claims 1, 17, and 20-22, recites "an expression evaluator that evaluates the one or more attributes . . . wherein the evaluation of at least one of the one or more attributes determines that a value to be shown for a field or property of the software application is the result of an evaluation of an expression contained in the code of the software application, which value is different from the value that would be shown for the field or property when the corresponding attribute is removed . . . ." *Bates* and *Bates_2*, considered both alone and in combination, do not teach this element.

Attributes are keyword-like tags that specify additional information about entities. (*See e.g.,* pg. 2, lns. 6-7 and pg. 6, lns. 7-8). A debugging system can refer to the information in the attribute to determine how objects should be used. (*See e.g.,* pg. 2, lns. 8-10 and pg. 6, lns. 8-11.) Attributed debugging relates to a debuggee (*e.g.,* process to be debugged) that includes attribute(s) that can be employed to facilitate debugging. (*See e.g.,* pg. 2, lns. 22-24.) The attribute definition declaratively indicates how the debug information is presented and since attributes are an extensible, declarative manner of conveying runtime information, the attributes allow that behavior to be specified. (*See e.g.,* pg. 2, ln. 25 to pg. 3, ln. 23.) Attributed debugging allows a developer of a type, who also understood and specified the runtime behavior of that type, to specify how the type will appear when it is being debugged. (*See e.g.,* pg. 6, lns. 18-20.)

In an example, attributed debugging allows the manipulation of the view of data in the debugger by allowing annotations that can be controlled. (*See e.g.,* pg. 2, lns. 30-31). This can facilitate control of a value that should be shown at a top-level, which can mitigate the need to expand the object for additional information. (*See e.g.,* pg. 3, lns. 1-2.) This can also facilitate control of whether a field or property should be presented, provide more descriptive information, and whether a type should be shown fully expanded. (*See e.g.,* pg. 3, lns. 5, 10, and 12-13). Further, this can include control of a value that should be presented for a field or property, which can be based on a result of an evaluation of an expression. (*See e.g.,* pg. 3, lns. 7-8, p.8, ln.16 – p. 9, ln. 7.)

Each of the independent claims has been amended to clarify these aspects of the claimed invention. Specifically, the addition of the language "an expression evaluator that evaluates the one or

more attributes . . . wherein the evaluation of at least one of the one or more attributes determines that a value to be shown for a field or property of the software application is the result of an evaluation of an expression contained in the code of the software application, which value is different from the value that would be shown for the field or property when the corresponding attribute is removed . . . ." in the independent claims clarifies that the expression evaluator causes information to displayed in the debugger that is different from the information that would be displayed absent the attribute. In other words, the evaluation of attributes by the expression evaluator results in the display of information that would otherwise not be displayed. *Bates* and *Bates_2*, considered both alone and in combination, do not teach this element.

Bates relates to providing descriptions of variables and/or providing comments or other information associated with the variables available while debugging. (*See e.g.,* pg. 1, ¶[0003] and ¶[0009] and pg. 2, ¶[0022] and ¶[00025].) An "attribute" for the variable can be set to "off" or "on". (*See e.g.,* pg. 6, ¶[0064] and Fig 3, elements 312 to 322 and corresponding text at pg. 4, ¶[0047].) These attributes are "machine-generated comments" or "use information" and includes a group of characters representative of a word or words, such as G(global), S(static), I(index), P(parameter), R(return), and C(call). (*See e.g.,* pg. 2, ¶[0026].) If an attribute is set to "on", the appropriate indicator is associated with the variable value. (*See e.g.,* pg. 6, ¶[0064].) Thus, *Bates* merely relates to a filtration system that allows certain data to be included or eliminated from view (by specifying whether that data is "on" or "off"). In other words, *Bates* merely allows the user to decide which characters will be displayed (those for which the data is "on") and which will not be displayed (those for which the data is "off"). However, *Bates* does not teach evaluating an expression contained in the code of the software application. Any evaluation by *Bates* is limited to evaluating whether the attribute is "on" or "off."

The secondary art, *Bates_2*, relates to a graphical debugger with a loadmap display manager and a custom record display manager that implements a custom record display function. (*See e.g.* col. 2, lns. 1-12.) A user can program the user interface to remember which fields of a variable type or variable are to be sent to an enhanced graphical user interface (GUI). (*See e.g.,* col. 3, lns. 49-52.)

When a variable of that type is encountered, the GUI initially displays only the user-programmed fields for the variable or the record. (*See e.g.*, col. 2, lns. 52-55.) Thus, *Bates_2* combined with *Bates.* merely teaches that characters for the user-selected fields will be displayed on GUI, the organization of which can be customized by the user. However, *Bates_2* in combination with *Bates* does not teach or suggest evaluating an attribute wherein the evaluation of the attribute determines that a value to be shown for a field or property of the software application is the result of ***an evaluation of an expression contained in the code*** of the software application, resulting in a value that is different from the value that would be shown for the field or property when the corresponding attribute is removed. In other words, Bates_2 in combination with Bates *et al.* does not teach the ability to evaluate an ***expression contained in the code.*** Thus, Applicant respectfully submits that independent claims 1, 17, and 20-22, and the claims depending therefrom, are allowable over *Bates* in view of *Bates_2.*

2. *Bates* and *Bates_2*, Considered Alone and In Combination, Do Not Teach An Attribute that Declaratively Indicates the Developer-customizable Format in which the Debug Information Is Displayed in a Developer-customizable Data Window, as Claimed in Claims 1-22

Each of the independent claims 1, 17, and 20-22, recites "the attribute definition declaratively indicates the developer-customizable format in which the debug information is displayed in a developer-customizable data window." *Bates* and *Bates_2*, considered both alone and in combination, do not teach this element as explained below.

As explained above, the attributes of the present invention can be used to control both the format of the information displayed in the debugger, and the arrangement of that information in the data window of the debugger. Independent claims 1, 17, and 20-22 have been amended to more-clearly capture this aspect of the invention. Specifically, the claims have been amended to clarify that the format of the information displayed is a different aspect than the organization of the information in the data window. For example, the DateTime is a representation of a particular moment in time. (Specification, at p. 6, lns. 18-31.) This information can be displayed in different representations, including a Julian date or a specific number of "ticks." (*Id.*) The attributes of the present invention

allow the developer to customize the format in which such information is displayed on the debugger. This is different from the mere organization of the information on the data window.

Bates and *Bates_2*, considered both alone and in combination, do not teach both a developer-customizable format and a developer-customizable data window. At best, *Bates_2* teaches that the data window of the debugger is developer customizable, such as by displaying only the user-programmed fields for a variable or record, as explained above. Because *Bates* in view of *Bates_2* does not teach a developer-customizable format, Applicant respectfully submits that claims 1-22 are allowable over these references.

3. *Bates* and *Bates_2*, Considered Alone and In Combination, Do Not Teach a DebuggerTypeProxy as Claimed in Claim 22

Independent claim 22 has been amended to recite, "wherein at least one of the attributes is a DebuggerTypeProxyAttribute that is employed to specify the display proxy of a type . . . ." There is no discussion in *Bates* and *Bates_2*, considered both alone and in combination, regarding a DebuggerTypeProxyAttribute. Consequently, Applicant respectfully submits that claim 22 is distinguished over *Bates* in view of *Bates_2*, and should, therefore, be allowed.

CONCLUSION

For the foregoing reasons, Appellant respectfully submits that the Examiner's rejections under § 103 are overcome, and requests that claims 1 -22 be allowed.

The Commissioner is hereby authorized to charge payment of any of the following fees that may be applicable to this communication, or credit any overpayment, to Deposit Account No. 23-3178: (1) any filing fees required under 37 CFR § 1.16; (2) any patent application and reexamination processing fees under 37 CFR § 1.17; and/or (3) any post issuance fees under 37 CFR § 1.20. In addition, if any additional extension of time is required, which has not otherwise been requested, please consider this a petition therefore and charge any additional fees that may be required to Deposit Account No. 23-3178.

Dated this 9th day of July, 2009.

Respectfully submitted,

/Chad E. Nydegger, Reg. # 61020/

RICK D. NYDEGGER
Registration No. 28,651
JENS C. JENKINS
Registration No. 44,803
CHAD E. NYDEGGER
Registration No. 61,020
BRIAN D. TUCKER
Registration No. 61,550
Attorneys for Applicant
Customer No. 47973

RDN:JCJ:CEN:BDT:hb
2402025_1.DOC